Cooking Game: Intelligent Animation

Elizabeth Jones

Graphics & Multimedia final project at **University of Sydney**

This course is a broad introduction to the field of graphics and multimedia computing.
Covers underpinning theories & practices in the field.
Emphasis was placed on the principles and cutting edge technique of ...

- multimedia data processing
- content analysis
- media retouching
- media coding and compression techniques

Video given:

Project Breakdown

01 Ideation

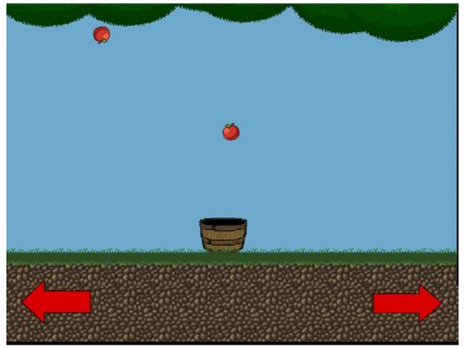
02 Extracting Frames

03 Applying Masks

04 Finding Centroids

05 Setting up my Game

06 Putting it all Together



Ideation

Took inspiration from the apple-catching arcade game

Characters

- Chef (main character)
- Cookie jar (user)
- Cookies (points)

Storyline

- During levels, the score is displayed in the top right corner and the timer is displayed in the top left corner (counts up in seconds)
- Level 1: Collecting three cookies before 30 seconds
- Level 2: The timer and score reset and players now have 30 seconds to collect five cookies.

Scene

- Start scene: The background is the basic kitchen with text at the bottom of the screen displaying “Ready to cook? (enter to start)”
- Two distinct kitchen settings, each representing a unique game level.
- Red or green screen based on the player’s success to end the game.

Extracting Frames

```
# Setup for object tracking

if not os.path.isdir(os.path.join(os.getcwd(), 'frames')):
    os.mkdir("frames")
else:
    print('frames already exists')

if not os.path.isdir(os.path.join(os.getcwd(), 'composite')):
    os.mkdir("composite")
else:
    print('composite already exists')

framenumber = 0
omovie = cv2.VideoCapture('Opt1-MarionetteMovements.mov')
frame_height = omovie.get(cv2.CAP_PROP_FRAME_HEIGHT)
frame_width = omovie.get(cv2.CAP_PROP_FRAME_WIDTH)

#Extract the frames from original video
while(1):
    ret, frame = omovie.read()
    if not ret:
        break
    print('Extracting: %d' % framenumber)
    clear_output(wait=True)
    cv2.imwrite('frames/%d.tif' % framenumber, frame)
    framenumber += 1
omovie.release()
```

Applying Masks

1. Color Masks

- Applied a red filter
- I created a lower bound rgb(0, 50, 150), and upper bound rgb(20, 255, 255) to best filter out everything in the frame beside the main, red points from the monkey
- Then using cv2.inRange I applied the mask and converted the new image, with my mask, to black and white.

2. Dilating Image

- Since my red mask kept some the monkey's face I needed to dilate the image
- This completely removed the outline of the face
- I dilated it 3 times, using a 3x3 kernel of 1s.

3. Eroding Image

- Points were disconnected, more than 5 groups, and very small
- I used a 6x6 kernel of 1s and eroded it twice.

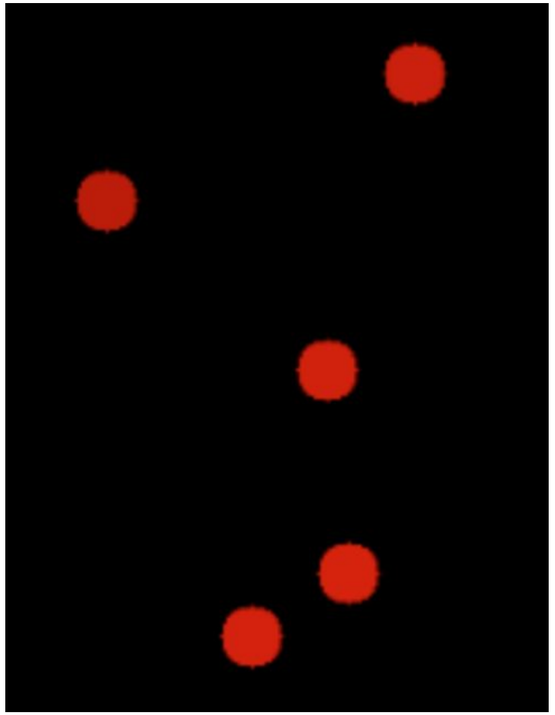
(a) Red Mask

(b) Dilated Image

(c) Eroded Image

Finding and Sorting Centroids

- Used contours from my final mask and put them through my helper function "labelBodyParts"
- Sorted through the centroids using PriorityQueue()
- Found the center of the marionette by going through each point in the given list and finding the maximum and minimum x and y values, then using them to find the approximated center
- Loop through each centroid and sort them based on how close they are to my approximated center
  - The first element of my PriorityQueue is my center.
  - Check if they are above or below my body, and positioned left or right and put in respective queue.
- PriorityQueues sort negative numbers by their largest to smallest absolute values, for my right hand and foot I subtracted the body x-value by the potential point to get the furthest point to be the highest priority.
- Created an array of length k, 5, and place the leftHand, rightHand, body, leftFoot, and rightFoot.
- “findCentorids()", I iterate through each frame and get the centroid points, I then check that all five centroid points exist, and if they do I append them to my final list of centroids.



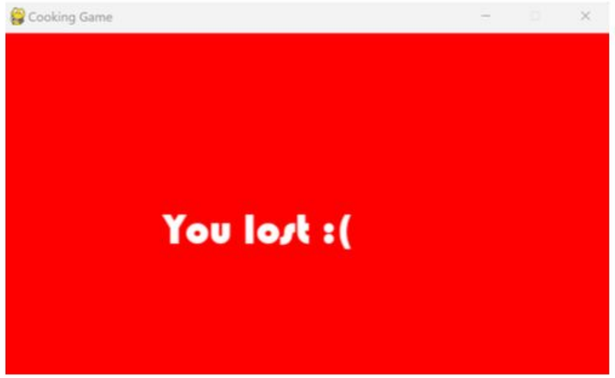
```python
findBody = queue.PriorityQueue()
for i in range(len(centroids)):
    dist = math.sqrt((centroids[i][0] - avgX)**2 + (centroids[i][1] - avgY)**2)
    findBody.put((dist, centroids[i]))
body = findBody.get(0)[1]

leftArm = None
rightArm = None
getArmLeft = queue.PriorityQueue()
getArmRight = queue.PriorityQueue()
for i in range(len(centroids)):
    if (centroids[i][1] < body[1]):
        if (centroids[i][0] < body[0]):
            getArmLeft.put((centroids[i][0] - body[0], centroids[i])) # problem because of negatives
        else:
            getArmRight.put((body[0] - centroids[i][0], centroids[i]))

if not getArmLeft.empty():
    leftArm = getArmLeft.get(0)[1]
if not getArmRight.empty():
    rightArm = getArmRight.get(0)[1]

leftLeg = None
rightLeg = None
getLegRight = queue.PriorityQueue()
getLegLeft = queue.PriorityQueue()
for i in range(len(centroids)):
    if (centroids[i][1] > body[1]):
        getLegRight.put((body[0] - centroids[i][0], centroids[i]))
        getLegLeft.put((centroids[i][0] - body[0], centroids[i]))

if not getLegRight.empty():
    rightLeg = getLegRight.get(0)[1]
if not getLegLeft.empty():
    leftLeg = getLegLeft.get(0)[1]

final_centroids = [leftArm, rightArm, body, leftLeg, rightLeg]
return final_centroids
```

Stages

- Start screen (e) is initial stage
- Once enter is clicked it brings the user to Level 1
  - Pass level 1 and move on
  - Fail -> failed screen
- If the user succeeds they move on to level 2, sleeker and more modern kitchen
  - Pass -> Succeeded Screen
  - Fail -> failed screen
- Both the success and failure screens are the final stage of the game.

(a) Level 1

(b) Level 2



(c) Failed Screen

(d) Succeeded Screen

(e) Start Screen

Next Steps

GameRules
- How will characters interact?
- Next level?

CookieJar
- Implement user handling keys
- How can it track when it caught a cookie or hit the chef?

BodyPart
- Chef hands, feet and center
- Need to be on the centroid points

Limbs
- Lines that draw from center to feet and hands
- Needs to update with centroids

Cookie
- Always at least one on the screen
- Drops from the chefs hands

CookingGame
- Putting it together with pygame

Implementing Characters

1. Chef

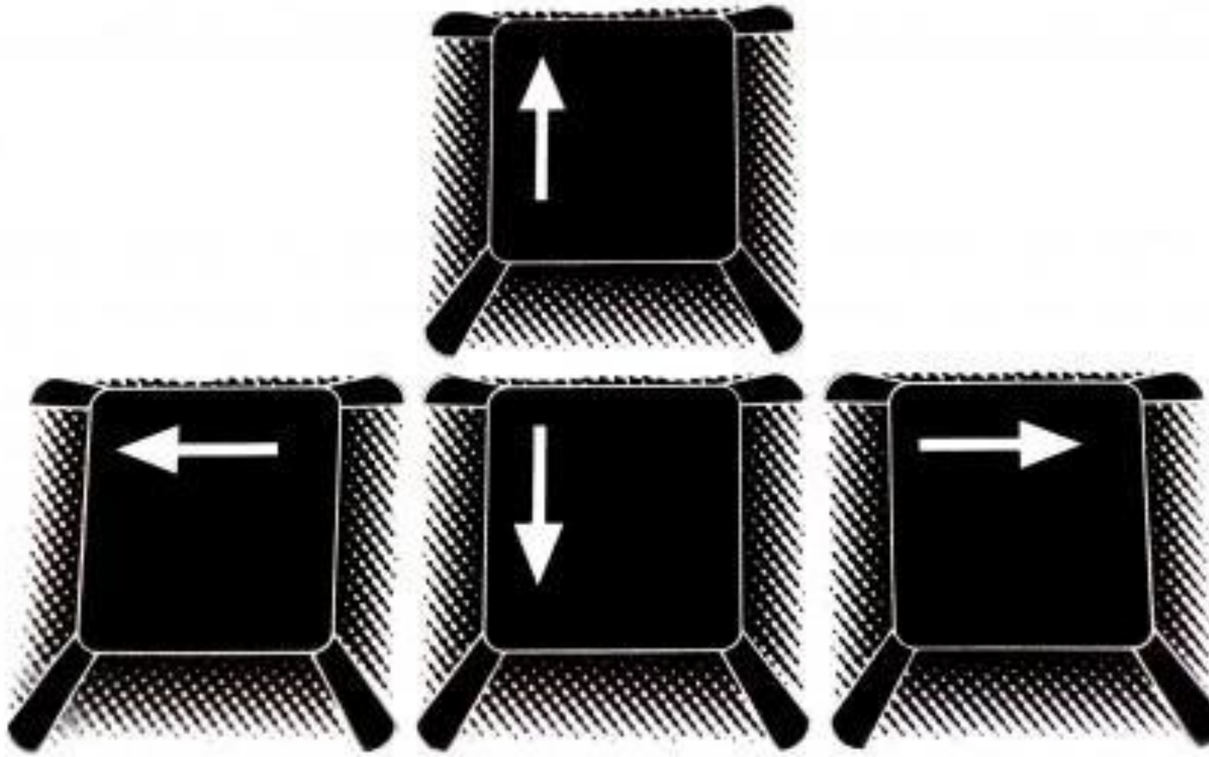
- Uses the centroids to mimic the monkeys movements
- To draw the centroids and switch between them at a speed of 30 frames per second, I used the pyGameClock, and an index for where I am on my centroid list
- Created a centroid switch time variable, 33 (TotalTime ÷ NumberofSwitches)
- While either level 1 or level 2 is running: check if the time since the last switch is greater than or equal to the centroid switch time
- Add the body parts from the index that I am currently on
- Draw the arms and legs using the limbs class, which draws lines from the hands/feet to the body centroid.

2. Cookie

The cookie falls from either the left or right hand at random
There is always one cookie one the screen at a time

3. Cookie Jar

The player moves up, down, left and right with the arrow keys.



Putting it all together

My main class, CookingGame, has four while loops that check different variables to determine what stage should be shown.

1. If the game has not started and the game is not over: Start screen
2. If the game has started and it is level one and the game has not been won: Level 1
3. If the game has started and it is not level one and level two has not been won: Level 2
4. If both games are over and both were won: Succeeded screen
5. If anything has been lost: Failed screen

Project Takeaways

- Apply multimedia data processing and analysis techniques to real world applications
- Understand data representation and enhancement for different building blocks
- Understand various multimedia data coding and retrieval techniques
- Obtain practical skills in graphics processing and analysis

Thank you!
Questions?